虚拟软件路由器数据包转发性能实验研究

贺鹏 杨建华 张建华 谢高岗

摘　要：未来网络的体系结构需要根据网络业务自适应进化调整，因此灵活与高性能的路由器是构建未来网络的基础。虚拟软件路由器支持网络虚拟化和可编程的特性，能够在一个物理网络上承载多个不同网络体系结构的逻辑网络，实现业务创新，具有良好的灵活性。但是虚拟化所带来开销会给虚拟路由器的数据包转发性能带来很大影响。如何在保证灵活性的前提下提高转发性能成为迫切需要解决的问题。本文对不同的虚拟技术和可扩展路由软件进行介绍，并搭建实验平台，测试了 KVM（全虚拟化）、Xen（半虚拟化）、OpenVZ（操作系统级别虚拟化）等的转发性能，评估了不同 I/O 加速技术对转发性能的提升。该实验数据对定位虚拟化软件路由器的转发性能瓶颈，提高转发性能具有重要的指导意义。

关键词：虚拟化 软件路由器 转发性能 I/O 加速

1　引言

近几年来，网络虚拟化成为一项被广泛关注的技术。虚拟路由器作为虚拟网络中的核心网络设备，对比现有网络设备有以下两点优势：1）虚拟路由器能够将一台物理路由设备虚拟成多台路由设备，多个虚拟设备并行处理不同的业务流量，并根据业务的特性提供相应的功能扩展，从而满足物联网、云计算服务等新兴应用对不同业务下的数据包处理策略提出的多样化要求。同时虚拟路由器还能根据虚拟网络的应用规模，给特定的虚拟设备分配特定的资源，尽可能提高设备的资源利用率。如，一个电信运营商可以使用单独一个虚拟网来承载一个大公司的网络业务，使用另一个虚拟网络来承载多个小公司的网络业务。各个虚拟网络完全隔离，提高了网络的鲁棒性。2）在一个由虚拟路由器搭建的虚拟网上，不同的虚拟设备可以支持不同的协议栈。如在一台虚拟设备中采用 TCP/IP 协议栈，而在另一台虚拟设备中完全采用新的协议。这样就能保证在同一个物理网络的基础设施上，并行进行不同的实验而不互相干扰，满足对未来互联网研究的需要。同时，相对于物理实验网络，虚拟实验网络能提供更快捷的搭建速度，更低廉的搭建成本。目前许多国家和组织正在积极研究未来互联网的发展，如美国的 GENI 项目[3]，欧盟的 FIRE 项目[4]等，提出了许多新的互联网架构。虚拟网能在不影响原有网络正常运行的条件下满足这些研究的实验需求，将这些先进的架构更快地部署到实际网络中。

现有的虚拟路由器实现平台可以分为两大类：通用硬件平台与专用硬件平台。通用硬件平台使用可扩展的路由器软件（如 Click[1]，XORP[2]等）和虚拟化软件（如 Xen[16]，OpenVZ[17]等）设计实现虚拟路由器。近年来，基于通用硬件平台的软件路由器在转发性能上取得了较大突破。多布雷斯库（M. Dobrescu）等[5]在充分挖掘单个物理设备的吞吐量潜力的基础上，基于集群方式设计实现了 RouteBricks 软件路由器。实验结果表明，四个通用硬件平台的服务器构建的 RouteBricks 集群，可以提供高达 35Gbps 的吞吐率。韩祥进（Sangjin Han）等[6]，利用 GPU[1]的强大并行计算能力，研制出单台吞吐率可达 39Gbps 的通用硬件路由器 PacketShader。专用硬件可编程平台的虚拟路由器使用专用硬件重新设计路由器的数据平面。如安瓦尔（M. B. Anwer）等[9]首次使用 NetFPGA[21]实现了多个虚拟路由器的数据平面。每

[1] Graphics Processing Unit，图形处理器

个数据平面采用相同的结构，可以实现查表，转发等功能。吕国翰（音译，Guohan Lu）等人[10]提出了一种可配置的数据转发引擎——CAFÉ。CAFÉ 通过使用不同 API[2]配置底层硬件，得到不同的数据平面，从而避免了重新设计硬件。乌尼克里希南（Deepak Unnikrishnan）等人[11]提出了硬件可重构和软件硬件迁移的概念，即当某个虚拟路由器需要处理的流量增大时，可以将该虚拟路由器的路由表拷贝到硬件上，实现硬件数据平面迁移，而将原先在硬件的数据平面迁移到软件层面；当需要部署新的数据平面时，则可以将所有的数据平面全部迁移到软件层面，对现场可编程门阵列写入新的逻辑。安瓦尔等人[12]提出了一种模块化的流水架构 SwitchBlade，支持硬件可编程，从而在保证硬件快速转发性能的前提下，支持新协议的快速原型设计与部署。使用专用硬件设计的虚拟路由器需要在保证高性能的同时，保证底层数据平面的可扩展性。

网络带宽的激增对路由器的吞吐率提出了非常高的要求。传统路由器对数据包的处理过程简单，在数据平面和控制平面上做了充分的优化，提供了非常高的吞吐率。虚拟软件路由器由于虚拟化、可扩展等技术特点，需要提供更高的转发性能，以满足各种新兴业务的深度数据包处理。在以上的讨论中，无论采用哪种设计方案，都需要使用到虚拟化技术，而虚拟化技术的引入势必会对路由器 I/O 效率产生影响。因此，对各种不同虚拟化技术所带来的性能开销进行评估是必要的。本文测量了几款有代表性的用虚拟化平台实现的软件路由器的转发性能，在此基础上评估了几种常见的 I/O 加速技术，以便定位转发性能瓶颈，优化转发性能。

本文其他章节安排如下，第二章主要对虚拟路由器的相关技术进行介绍，包括不同的虚拟技术的原理，几种可扩展路由器的架构等。第三章对几种常见的虚拟软件的数据包转发性能进行了测试，并给出实验结果。第四章介绍了一些常见的 I/O 加速技术，并搭建和评估了这些 I/O 加速技术。第五章对全文进行总结，并对 I/O 加速研究的可能途径进行了展望。

2 虚拟化软件路由器原理与实现

2.1 虚拟化软件路由器

虚拟软件路由器的典型框架如图 1 所示，虚拟路由器由三大组件构成：虚拟化软件平台，数据转发模块，控制模块。虚拟软件作为中间层（Hypervisor），给不同虚拟机提供虚拟 I/O 通道。数据转发模块可以由专用硬件或者通用网卡构成。在虚拟机内部则可以运行控制模块计算路由，或者用于承载实验网，进行新协议（如 OpenFlow[7]等）的实现。

图1. 虚拟路由器框架图

2.2 虚拟化实现

虚拟软件路由器所使用的虚拟化技术按照虚拟层次的不同，大致可以分为三类

[2] Application Programming Interfaces，应用程序接口

■ **全虚拟化（Full Virtualization）**

全虚拟化又称本地虚拟化（Native Virtualization）。该技术在操作系统和底层硬件之间使用一个中间层 Hypervisor，使得底层硬件能够被上层操作系统共享访问。当客户操作系统执行某项特权指令时，中间层将接管该段代码，并执行对应的操作，如给某个硬件的某个寄存器写入一个值等。由于在全虚拟化技术中，客户操作系统不能对底层硬件进行任何直接控制，而必须通过中间层进行代理，使得 I/O 性能会受到一定影响。相对于硬件虚拟化技术，全虚拟化技术可以支持不同的操作系统，但是不能模拟不同硬件平台。全虚拟化技术的代表软件为 VMware，KVM[14]等。近年来一些新的 X86 系列的 CPU 在指令级别上支持全虚拟化，使得全虚拟化环境下的操作系统能达到原生级别的运行速度。

■ **半虚拟化（ParaVirtualization）**

半虚拟化技术又称准虚拟化技术。该技术同样采用中间层来隔离底层硬件和操作系统。不同的是，它试图修改操作系统代码，让客户操作系统本身知道自己运行在一个虚拟化平台上。该技术能够显著提升虚拟化技术的 I/O 性能。例如，当网卡收到一个数据包时，在采用全虚拟化的情况下，中间层必须要模拟硬件中断，然后将数据包传输到某个操作系统上，而由于在半虚拟化技术中，操作系统本身就知道自己运行在一个虚拟化平台上，底层的中间层可以使用一定方式，批量地将数据包传输给该操作系统，而不需要模拟硬件中断。由于半虚拟化技术需要对操作系统代码进行修改，使得该项技术在非开源的商业操作系统上应用比较困难，如 Windows。半虚拟化技术代表软件为 Xen[16]，UML[15]等。

■ **操作系统级别虚拟化（Operating System-level virtualization）**

操作系统级别虚拟化和以上两种虚拟化都不相同，更类似于 UNIX 的 Jail 系统调用。它利用操作系统所提供一些机制（如 Linux 中命名空间），将一组进程放入到“容器”（container）中，与主操作系统的进程隔离开来，并使用这些机制限制这一组进程的 CPU 和内存占用率，从而达到虚拟化技术中资源的隔离与分配的目的。这类虚拟化技术只能运行在特定的操作系统上（几乎都是 Linux）。所有的“虚拟机”（在该技术中，称为 Server）都共享一个内核。该虚拟化技术的代表软件为 OpenVZ[17]，Linux-VServer[18]，LXC[19]等。操作系统级别虚拟化所带来的开销非常小。但其主要缺点是不能运行多个异构的操作系统。

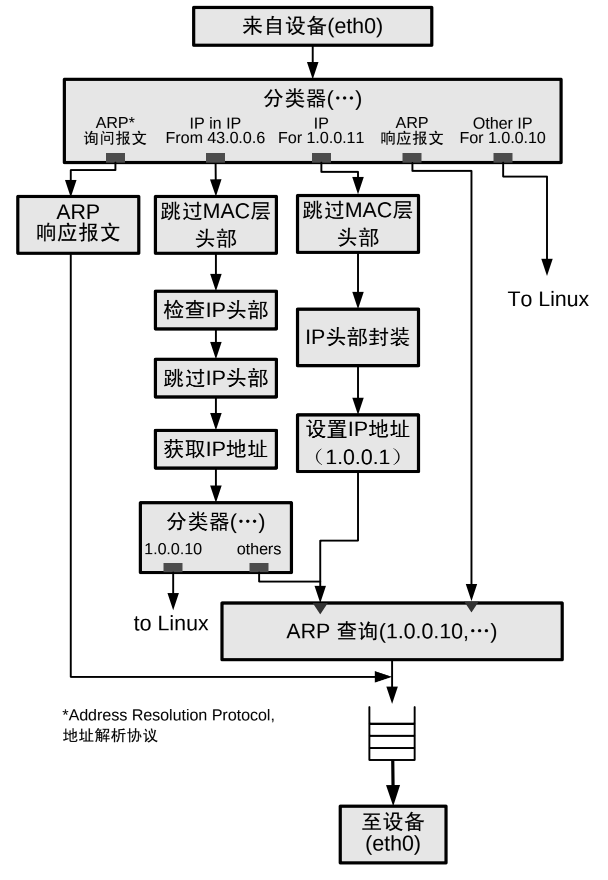


图2. Click 以太网交换机（Ethernet Switch） 配置图

2.3 数据转发模块

(1). Click

Click 是一个模块化的软件路由器。它将路由器设计的各个部分模块化（Click 称之为“元素”Element），提供了一种灵活的可扩展的路由器软件套件。Click 抽象出两种操作，将不同元素连接起来。一种称为 Push，即数据包由源端处理完毕后，从该连接的源端传递到目的端；一种称为 Pull，即连接的目的端发起一个请求，向源端请求一个数据包处理。图 2 是一个使用 Click 组合成的以太网交换机的框架图。不过，Click 虽然能够将数据包转发到指定的端口，但是并没有提供路由计算的能力。Click 不能动态地计算路由。

Click 软件套件对 Linux 内核进行了修改，提供了轮询驱动方式，能够在普通网卡上获得小包转发性能的很大提升。许多关于软件路由器的研究[5,8]使用了 Click 作为他们的数据转发模块。一些路由器软件，如 XORP 等，也可以将 Click 作为其转发引擎。RouteBricks 正是使用了 Click 的轮询驱动达到了非常可观的吞吐率。

(2). 硬件数据平面

使用专用硬件构建路由器的转发模块，在数据包转发性能上有软件不可比拟的优势，并且这些硬件在设计上还集成了路由表。使用 TCAM[3]等硬件进行快速路由查找，能够大大降低 CPU 的负担。不足之处在于，硬件的资源十分有限，目前容量最大的 TCAM 也只能存储几千条路由表，不能满足实际网络中 20~30 万的表项存储需求。使用硬件作为数据转发模块需要设计算法，构建“快表”，将频繁查找的 IP 表项交由硬件处理，而不频繁的表项交由软件处理。

2.4 控制模块

(1). 基于路由的控制模块 XORP

汉德利（M. Handley）等人意识到现有的商业路由器软件的封闭导致新协议的实现和研究缺乏一个可靠的测试平台，因此设计并实现了 XORP，旨在提供一个开放的、可扩展的路由器软件平台，便于研究人员部署和实现新路由协议。XORP 已经被工业界和教育研究机构广泛采纳。

XORP 采用多进程机制，通过转发引擎抽象层（Forward Engine Abstract）将数据包的交换及路由计算和转发引擎隔离开来，既保证了鲁棒性，也保证了开放性和性能。图 3 是 XORP 的结构示意图。图中 XORP 使用了 Click 作为转发引擎。

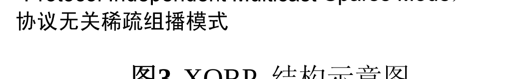


图3. XORP 结构示意图

(2). 基于转发表的控制模块 OpenFlow[7]

OpenFlow 是斯坦福大学提出的一种新的网络体系结构。OpenFlow 采用主从结构来进行网络管理。在一个 OpenFlow 网络中，

[3] Ternary Content Addressable Memory，三态内容可寻址存储器

主控设备（Controller）存储所有的转发策略，从设备（OpenFlow Switch，交换单元）则负责数据转发。每个从设备都包含一张转发表，存储转发规则。当数据包匹配其中一项规则，从设备便按照规则将数据包从它的某个端口转发出去，当数据包没有匹配其中任何一项，从设备便咨询主控设备，获得新的转发规则。OpenFlow 使用“十元组”（MAC[4]对，IP 对，端口对，等等）作为转发策略依据，能够实现非常复杂的转发策略。同时，主控设备开放了操作转发表的 API，能满足可编程网络的需求。

3 不同虚拟平台的转发性能测试

我们根据虚拟化层次的不同，选取了三款主流虚拟化软件，KVM（全虚拟化），Xen（半虚拟化），OpenVZ（操作系统级别虚拟化），分别进行了数据包转发测试。实验平台的各项参数如表 1 所示。

表 1 实验平台参数

硬件	规格说明
CPU	Intel Xeon L5420 2 路四核，共 8 核 每核一级缓存 32K:16K（指令）+ 16K（数据） 四核两两共享二级缓存 6M
内存	DDR2 667M 赫兹 16G
主板	SuperMicro X7DWU
网卡	Intel 82571EB 千兆，四口

3.1 实验配置

我们使用 Spirent Testcenter 作为测试仪表，采用 RFC 2544 标准测试方式测试了不同的虚拟软件的吞吐率。选取的包大小依次为 64 字节（最小包）、512 字节（网络中的平均包大小）、1518 字节（最大包）。同时，我们根据上述实验结果，又设计了多虚拟机情况下的数据包吞吐率实验。我们将不同的虚拟机运行在不同的网段中测试其总体转发性能，由于 Spirent 仪表不支持这种测试方式，我们采用了另一种方式表征吞吐率，具体说明见 3.2.2 节。

不同的虚拟软件的网络配置方式各有不同，我们只选取桥接（Bridge）模式的网络配置来进行。以下对这些软件做一些简单的介绍，并分别说明不同虚拟软件的网络配置情况。

(1). KVM

KVM，意即内核虚拟机（Kernel Virtual Machine），与主流虚拟化技术 Xen 的最大区别在于，它并没有单独实现一个中间层，而是向内核添加一两个模块，将内核作为中间层的一种虚拟化技术。KVM 是全虚拟化技术，需要 CPU 对虚拟化有一定的支持。图 4 显示了 KVM 的网络配置。图中 eth ***x*** 为以太网接口；tap ***x*** 为客户操作系统的以太网网口在中间层上的一个镜像网口；Br ***x*** 为桥接模块。

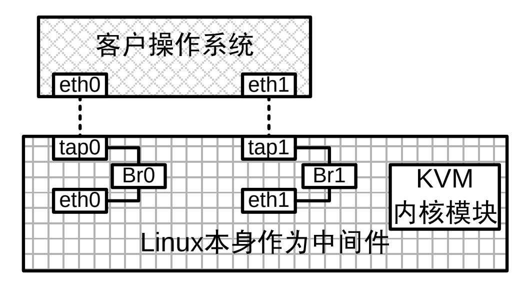


图4. KVM 网络配置

(2). Xen

Xen 是剑桥大学开发的一套半虚拟化软件，它实现了一个精简的中间层 Hypervisor。Xen 使用术语 Domain（域）来描述虚拟化环境。Xen 具有两种 Domain，Dom0 和 DomU。Dom0 又称 Driver Domain（驱动域），运行在该 Domain 上的操作系统能够直接访问硬件。运行在

[4] Media Access Controller，媒体存储控制器

DomU（客户域）的操作系统则必须通过 Dom0 的代理。测试中 Xen 的网络配置采用桥接模式，分别测试了 Dom0 和 DomU 的转发性能。图 5 显示了 Xen 的 DomU 网络配置情况。图中 peth ***x*** 为实际物理网口；vif ***x*** 为客户操作系统的以太网网口在中间层上的一个镜像网口 。

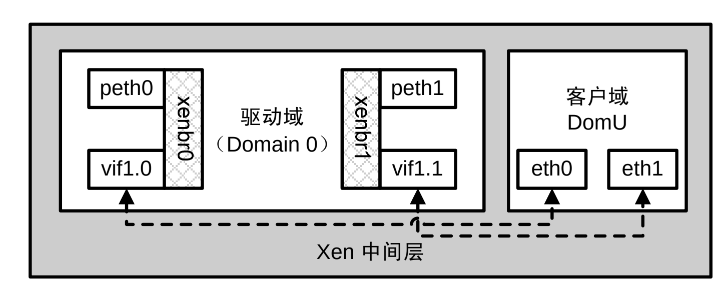


图**5.** Xen DomU 网络配置

(3). OpenVZ

OpenVZ 是 Swsoft 公司专有软件 Virtuozzo 的开源版本。它使用一种虚拟网络接口对的方式实现桥接。图 6 显示了 OpenVZ 的网络配置情况。图中 veth ***x*** 为客户操作系统的以太网网口在中间层上的一个镜像网口。

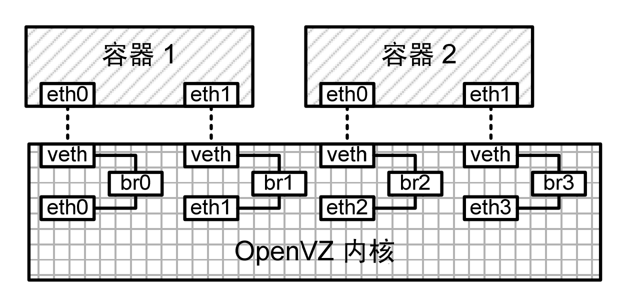


图**6.** OpenVZ 网络配置

3.2 实验结果与分析

3.2.1 原生操作系统 Native Linux 测试

我们首先对没有采用任何虚拟软件的系统平台（Native Linux）进行了数据包转发测试，测试结果如下。从测试结果可以看出，普通网卡对最小包的转发性能较差。这主要是由于普通网卡采取的中断方式所带来的系统开销过大。一些研究表明[5,6]，采取轮询方式，对小包转发能够带来很大的性能提升。

表 2 Native Linux 转发性能测试

帧长度	速率(%)	速率(帧/秒)	理论最高速率（帧/秒）
64	26.171	778,898	2,976,190
512	100	469,924	469,924
1518	100	162,548	162,548

3.2.2 虚拟化技术性能测试结果与分析

表 3 各种虚拟化技术转发性能测试

帧长度	吞吐速率（帧/秒）			
	KVM	Xen Dom0	Xen DomU	OpenVZ
64	29,762	611,488	113,452	590,565
512	31,132	469,924	139,506	469,924
1518	26,541	162,548	124,831	162,548

表 3 是分别对不同虚拟化软件的数据包转发性能测试后得出的结果。从结果中可以看出，全虚拟化带来的 I/O 性能开销非常大。这主要是由于数据包在内核中经过的路径太长，上下文切换过多。而半虚拟化则由于其针对虚拟化环境做了一定的优化，性能上相对于全虚

拟化有一定的改进，但是仍然和原生操作系统的性能有较大差距，而操作系统级别的虚拟化则由于开销非常小，除小包转发外，在各项数据上都和原生操作系统非常接近。

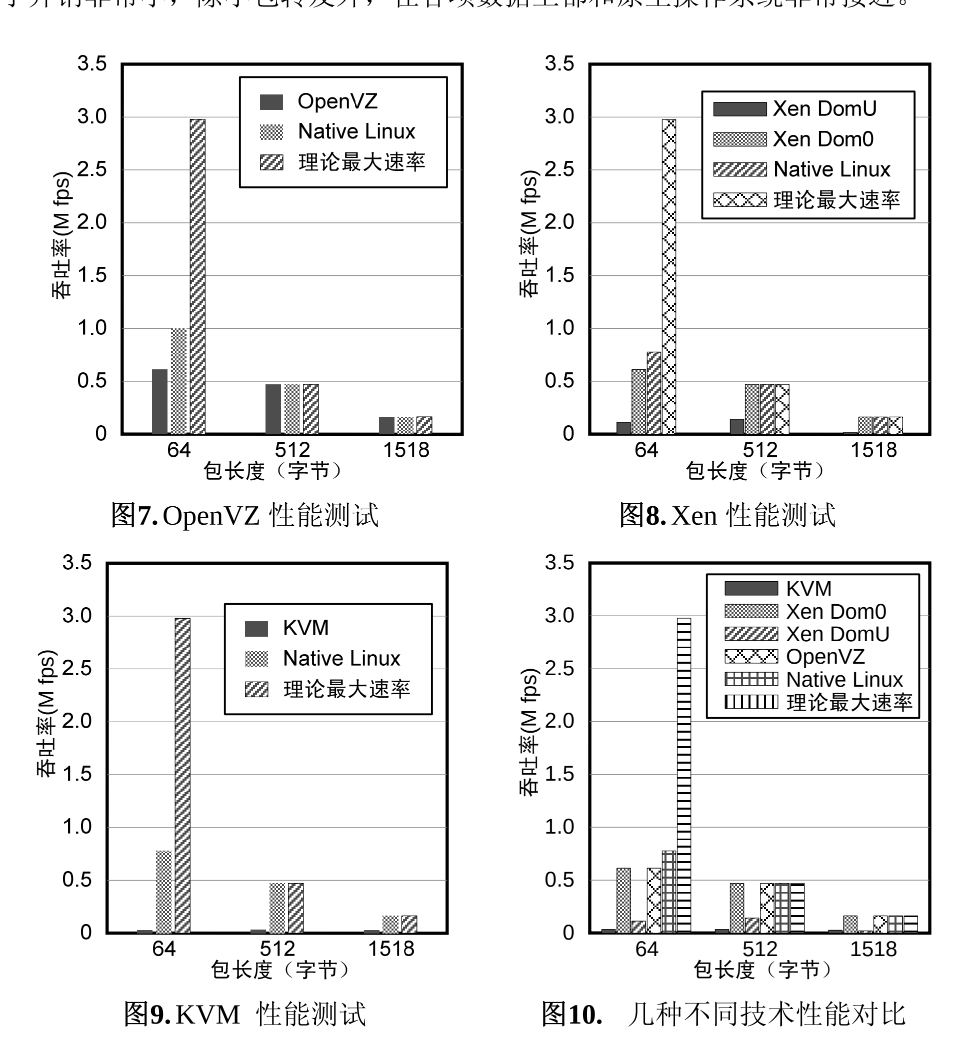


图7. OpenVZ 性能测试

图8. Xen 性能测试

图9. KVM 性能测试

图10. 几种不同技术性能对比

根据实验结果，我们认为，全虚拟化技术由于虚拟化开销太大，不适合作为虚拟路由器的虚拟化平台。半虚拟化和操作系统级别虚拟化各有优缺点。半虚拟化技术虽然在转发性能上与原生系统相比有很大差距，但它提供了更好的隔离性和鲁棒性。根据实验结果，我们又对 Xen 和 OpenVZ 进行了多虚拟机情况下的性能测试，结果如表 4 所示。表 4 使用实际转发速率（fps）与理论最大转发速率的百分比表示性能。

表 4 不同虚拟机的转发性能

虚拟机	帧长度（字节）	虚拟机个数 1	2	4	8	Native Linux
Xen	64	1.20%	6.12%	7.54%	8.05%	26.171%
	512	14.35%	22.73%	29.70%	35.10%	100%
	1518	74.21%	69.62%	60.2%	61.16%	100%
Open VZ	64	10.48%	10.39%	9.53%	12.66%	26.171%
	512	54.50%	53.84%	52.50%	53.00%	100%
	1518	100%	100%	100%	100%	100%

从实验结果得出，虚拟机数量的不同对于吞吐量的影响并不大。由此可见，提高吞吐率的途径在于缩短数据包的内核路径，给虚拟机直接访问设备的能力。现有一些方法可以显著

提升虚拟 I/O 的性能，将在下一章主要介绍。

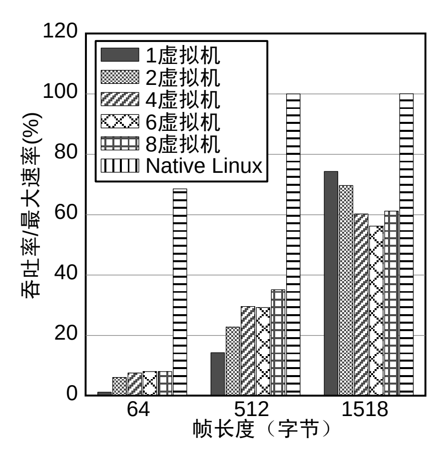


图11. Xen 多虚拟机转发性能

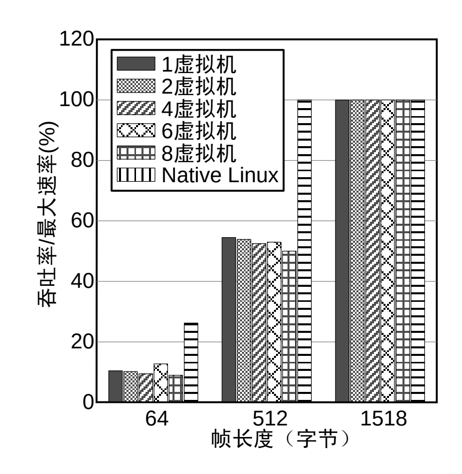


图12. OpenVZ 多虚拟机转发性能

4 虚拟 I/O 加速技术

4.1 I/O 性能的加速技术

上述测试的实验结果可以看出，全虚拟化技术所带来的开销会对 I/O 操作的性能带来影响。但是全虚拟化技术可以让操作系统无需修改就可以运行，具有很高的适用性。研究人员对此进行研究，设计出了一些能够显著提升虚拟 I/O 性能的加速技术。以下分别加以介绍。

(1). Virtio(半虚拟化 I/O)[20]

Virtio 技术是澳大利亚程序员拉塞尔（R. Russell）设计并实现的，旨在给 Linux 操作系统中越来越多的虚拟化软件提供一套标准化的半虚拟 I/O 接口。半虚拟化 I/O 能够让客户操作系统意识到自身运行在虚拟化环境中，并在这个基础上采用一定的优化措施，提升客户操作系统的 I/O 性能。图 13 是半虚拟化 I/O 的原理图。目前，KVM 支持该虚拟 I/O 加速技术。

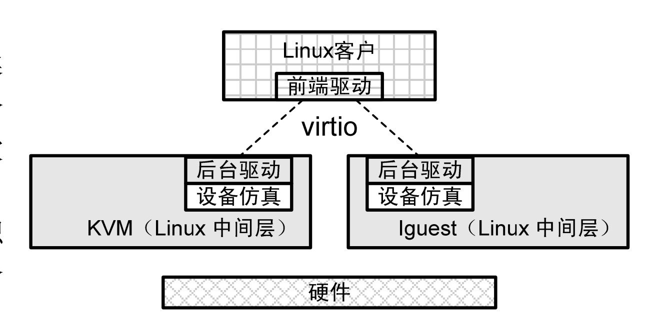


图13. 半虚拟化 I/O 技术原理

(2). Intel VT-d 技术与 SR-IOV 技术

尽管半虚拟化 I/O 技术能够提升客户操作系统的 I/O 性能，但仍然存在上下文开销过高等问题。解决问题的关键在于取消中间层的代理，让客户操作系统能够直接访问到物理设备。具体说来，直接访问物理设备可以划分为两个问题：1）对客户操作系统的设备缓冲区内存直接访问（DMA Remapping）。2）中断重映射。Intel VT-d 技术（Intel Virtualization Technology for Directed I/O）解决了以上两个问题。如图 14 所示：图中左半部表示的是全虚拟化 I/O，需要由中间层代理，客户操作系统才能访问底层硬件，而右图中由于 DMA Remapping 的存在，底层硬件能够直接将数据写到客户操作系统的缓冲区内，使得 I/O 性能接近原生操作系统的性能。

VT-d 技术提供了数据由底层硬件到客户操作系统的直接通路，SR-IOV（Single-Root I/O Virtualization，单根读写虚拟化）则提供了将单个设备虚拟成多个设备的方法。SR-IOV 为每

个虚拟机提供独立内存空间、中断资源。SR-IOV 引入了两个新的功能类型：物理功能（Physical Functions, PFs）和虚拟功能（Virtual Functions，VFs）。

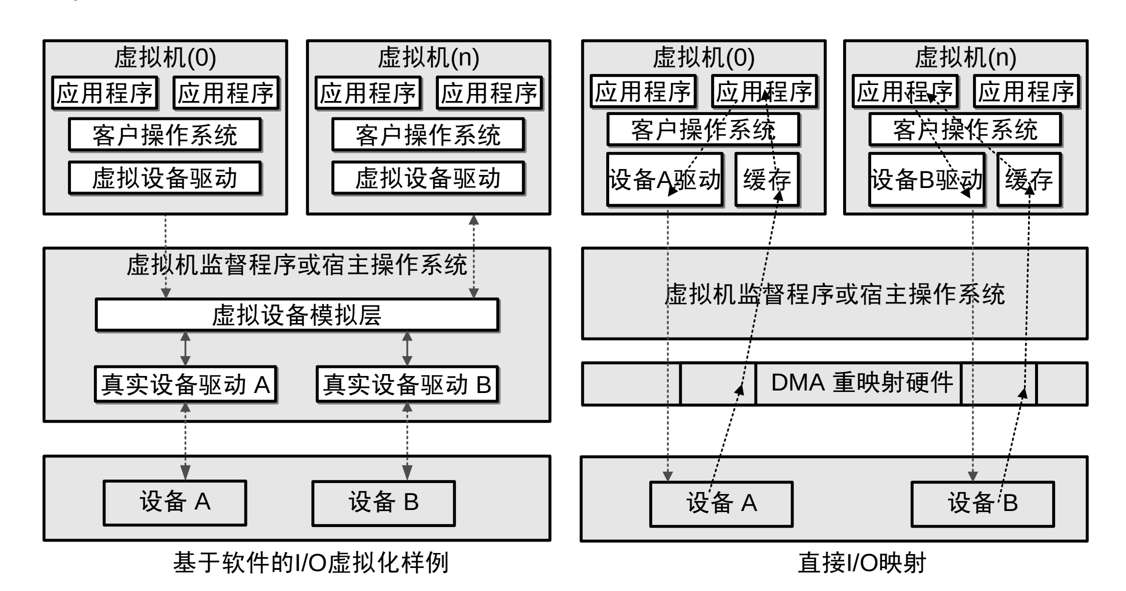


图14. 对客户操作系统的设备缓冲区内存直接访问

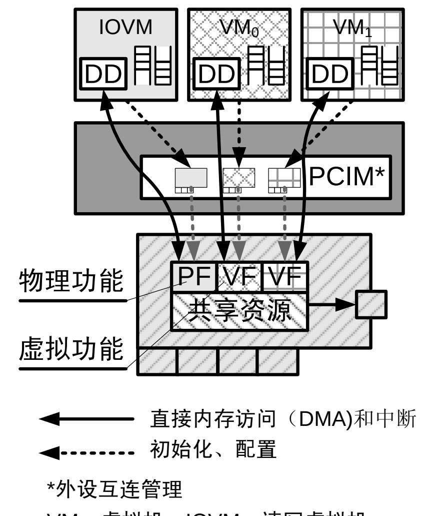


图15. SR-IOV 架构示意图

- 物理功能（Physical Functions，PFs）这是一些支持 SR-IOV 扩展功能的 PCIe[5]功能，被用于配置和管理 SR-IOV 功能特性。
- 虚拟功能（Virtual Functions，VFs）这是一些“精简”的 PCIe 功能，包括数据迁移必需的资源，以及经过谨慎精简配置的资源集。

SR-IOV 架构的设计允许一个 I/O 设备支持多个虚拟功能。从而提供了一种不需要软件模拟就可以共享 I/O 设备和 I/O 端口的物理功能的方法。图 15 是 SR-IOV 的架构示意图。

(3). PCI Passthrough 技术

PCI Passthrough 是 Xen 开发人员开发的客户虚拟操作系统直接访问设备的技术。主要原理是利用软件模拟 IOMMU[6]，使得客户操作系统能越过中间层访问设备。和 VT-d 技术相比，PCI Passthrough 的优势在于，前者是一种从硬件上支持 IOMMU 的技术，尚未广泛普及，而后者则从软件层面支持了 IOMMU，可以为不支持 VT-d 的系统进行 I/O 性能优化。

(4). 软件路由器的一些设计经验

- **RouteBricks 设计经验** RouteBricks 主要使用三种手段优化了单个路由器的转发性能：第一，使用轮询方式代替传统硬件的中断方式，极大地提高了通用硬件对 64 字节小包的转发性能；第二，采用批处理的方式，每次总线事务（bus transaction）都传递多个数据包的地址，从而充分利用了 PCIe 的总线带宽；第三，利用网卡的多队列特性，将不同队列与不同核进行绑定，极大地提升了数据处理的并行性。使

5 Peripheral Component Interconnect Express，新一代的 PCI 总线接口
6 input/output memory management unit，读写内存管理单元

用以上三种手段，可以使得单台服务器的 64 字节小包转发速率达到 9.77Gbps。

- **PacketShader 设计经验** PacketShader 对 Linux 的内核协议栈进行了优化，将内核中表示数据包的结构体 sk_buff 由原来的 208 字节减少为 8 字节，采用批处理的方式，极大地提升了数据包处理性能。实验表明，仅采用这种优化手段，可以将数据包吞吐量提升到 10.5Gbps，性能提升 13.5 倍。PacketShader 同时也利用了多核多队列的特性，并且考虑到其采用的架构为 NUMA(非一致性内存访问)，针对数据包的分发均衡性和提升数据局部性做了精心优化，使其整体转发性能达到 40Gbps。

4.2 虚拟 I/O 加速技术性能评估

本节对 VT-d 技术，半虚拟化 I/O 技术，PCI Passthrough 技术所带来的 I/O 性能提升进行评估。使用了 RFC2544 标准吞吐率测试方法进行了测试。此外，对半虚拟化 I/O 还进行了 IPerf[22]带宽测试。由于 Xen 不支持 Virtio，我们的实验平台上也无法为 Xen 打开 VT-d 的技术支持，我们最终在 KVM 上实验了 VT-d 技术和 Virtio 的技术，在 Xen 上测试了 PCI Passthrough 技术。

表 5 KVM 测试结果

帧长度	吞吐率（帧/秒）	
	KVM virtio	**KVM VT-d**
64	113,452	841,696
512	89,943	413,750
1518	69,970	162,548

表 6 PCI Passthrough 测试结果

帧长度	吞吐率（帧/秒）	
	Xen DomU	**Xen DomU+ PCI passthrough**
64	113,452	230,650
512	139,506	222,110
1518	124,831	162,548

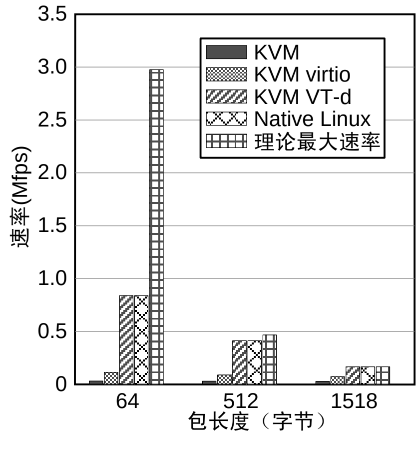


图16. 16 KVM I/O 加速

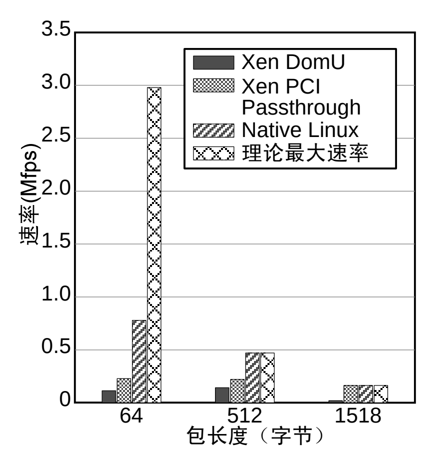


图17. PCI Passthrough 软件模拟

表 7 KVM 半虚拟化 I/O 的 IPerf 带宽测试

	KVM		**KVM virtio**	
	单向	双向	单向	双向
带宽	210Mbps	117Mbps 332Mbps	749Mbps	555Mbps 517Mbps

实验结果表明，半虚拟化 I/O 技术对于客户操作系统来说，确实能提高 TCP 传输的带宽，但是对于数据包转发，性能提升则是有限的。VT-d 技术则给客户操作系统直接访问设备硬件的能力，能够获得接近原生操作系统的转发性能，将虚拟化所带来的开销降至最低。

PCI Passthrough 技术由于软件模拟的开销，所提供的性能则在两者之间。

5 结束语

本文围绕虚拟路由器这一概念分别探讨了几种虚拟化技术、与之相关的 I/O 加速技术以及业界广泛认可几种可扩展路由器软件，并在此基础上，测量了几款具有代表意义的虚拟化软件的转发性能，对虚拟路由器的研究和设计具有启发意义。实验结果表明，操作系统级别的虚拟化技术所引入的额外开销最小，然而该项技术由于其虚拟化程度较低，使用起来有一定的局限性。采用 VT-d 等硬件支持的虚拟化技术能够使运行在全虚拟化环境下的客户操作系统获得原生的 I/O 效率，同时又保持了虚拟化层次高的优势。虽然现有的虚拟路由器的设计均是采用 OpenVZ 等轻量级虚拟化技术，但可以预期，VT-d 和 SR-IOV 技术将会对虚拟路由器的设计带来较大影响。

在目前的研究中，并没有使用通用平台搭建的虚拟路由器，大部分已有研究均使用专用硬件来降低虚拟化开销，并采用 OpenVZ 等轻量级虚拟系统来进一步降低虚拟化开销。然而，由于 SR-IOV 等技术的出现，虚拟化开销已经被降至最低，因此一些软件路由器提高 I/O 性能的经验完全可以借鉴，用来设计虚拟路由器。可以预见，未来一到两年内，会出现使用 SR-IOV 技术的软件虚拟路由器的研究。

参考文献：

[1] Robert Morris, et al. The Click modular router. SOSP, 1999

[2] Mark Handley, et al. XORP: An Open Platform for Network Research. ACM SIGCOMM Computer Communication Review, 2003

[3] GENI. http://www.geni.net/

[4] FIRE. http://cordis.europa.eu/fp7/ict/fire/

[5] MiHai Dobrescu, et al. RouteBricks: Exploiting Parallelism To Scale Software Routers. SOSP 2008. Best Paper Awards.

[6] Sangin Han, et al. PacketShader: a GPU-Accelerated Software Router. SIGCOMM 2010.

[7] Nick McKeown, et al. OpenFlow: Enabling Innovation in College Networks. ACM SIGCOMM Computer Communication Review, 2008

[8] Nobert Egi, et al. Towards High Performance Virtual Routers on Commodity Hardware. ACM CoNEXT, 2008.

[9] Muhammad Bilal Anwer, et al. Building a Fast, Virtualized Data Plane with Programmable Hardware. VISA 2009.

[10] Guohan Lu, et al. CAFÉ: A Configurable pAcket Forwarding Engine for Data Center Networks. PRESTO’09.

[11] Deepak Unnikrishnan, et al. Scalable Network Virtualization Using FPGAs. FPGA’10

[12] Muhammad Bilal Anwer, et al. SwitchBlade: A Platform for Rapid Deployment of Network Protocols on Programmable Hardware. SIGCOMM 2010.

[13] QEMU. http://wiki.qemu.org/

[14] Kernel Virutal Machine. KVM. http://www.linux-kvm.org/

（下转第 32 页）